# Features

With the reverse proxy configuration that was elaborated on in the previous chapter, the presence or absence of specific features wasn't much of a problem. This is because Nginx would simply have to differentiate between static and dynamic content, and consequently, serve the static file requests and forward the dynamic file requests to a backend server.

However, when you start considering Nginx as a possible full replacement for your current web server, you better make sure that you know what's in the box. If your projected architecture requires specific components, the first thing you would usually do is check the application features. The following table lists a few of the major features, and describes how Nginx performs in comparison to Apache.

## Core and functioning

The features of Nginx and Apache are shown in the following table:

| Features | Nginx | Apache |
|---|---|---|
| **HTTP request management** (how the web server processes client requests) | **Event-driven architecture**: In this architecture, requests are accepted using asynchronous sockets, and aren't processed in separate threads to reduce memory and CPU overhead. | **Synchronous sockets**, **threads**, **and processes**: In this, each request is in a separate thread or process and uses synchronous sockets. |
| **Programming language** (language that the web server is written in) | **C:** The C language is notably low-level and offers more accurate memory management. | **C** and **C++**: Although Apache was written in C, many modules were designed with C++. |
| **Portability** (operating systems that are currently supported) | **Multiplatform**: Nginx runs under Windows, GNU/Linux, Unix, BSD, Mac OS X, and Solaris. | **Multiplatform**: Apache runs under Windows, GNU/Linux, Unix, BSD, Mac OS X, Solaris, Novell NetWare, OS/2, TPF, OpenVMS, eCS, AIX, z/OS, HP-UX, and more. |
| **Year of birth** | **2002**: While Nginx is younger than Apache, it was intended for a more modern era. | **1994**: Apache is one of the numerous open source projects initiated in the 90s that contributed to making the World Wide Web what it is today. |

# General functionality

This section mainly focuses on the differences between Apache and Nginx rather than listing all sorts of features that have already been covered in the previous chapters:

| Feature | Nginx | Apache |
|---|---|---|
| **HTTPS support**: This specifies whether the web server can deliver secure web pages. | **Supported as a module**: If you require HTTPS support, you need to make sure that you compile Nginx with the proper module. | **Supported as a module**: Apache comes with HTTPS support via a module included by default. |
| **Virtual hosting**: This specifies whether the web server can host multiple websites on the same computer. | **Supported natively**: Nginx natively supports virtual hosting, but is not configured by default to accept per-virtual-host configuration files (more details are provided further on in this chapter). | **Supported natively**: Apache natively supports virtual hosting and offers the possibility to include one configuration file per folder (`.htaccess`). |
| **CGI Support**: This specifies whether the web server supports CGI-based protocols. | **FastCGI**, **uWSGI**, **SCGI**: Nginx supports FastCGI, uWSGI, and SCGI (as well as HTTP proxying) via modules that are included by default at compile time. | **CGI**, **FastCGI**: Most CGI protocols are exploitable via modules that can be loaded into Apache. |
| **Module system**: This specifies how the web server handles the modules. | **Static module system**: Modules are built-in; they must be included at compile time. | **Dynamic module system**: Modules can be loaded and unloaded dynamically from configuration files. |

Generally speaking, Apache has a lot to offer, notably a much larger number of modules available. Most of its functionality is modularized, including its core engine. At this time, the official Apache module website references over 500 modules for various version branches, compared to a little more than 100 for Nginx.

# Flexibility and community

This is another criterion for establishing an honest comparison between two applications of the likes of Nginx and Apache. In today's information technology industry, you cannot simply look at the raw functionality of a server application without considering questions such as:

- Where am I going to get help if I get stuck?